06

Cuenta abstracta

Transcripción

[00:00] Ahora que conocemos los beneficios de las clases abstractas, vamos a practicar un poco más. Esta vez, de la misma forma que hicimos con funcionario, lo vamos a hacer con la clase cuenta. ¿Por qué? Porque conceptualmente hablando vamos a repetir un poco los conceptos. Cuenta es un concepto muy genérico digamos, tú no vas al banco a decir solamente "Ábrame una cuenta".

[00:30] Y ellos tampoco te van a decir: "Okay, te voy a abrir solamente una cuenta". Lo que te van a decir es: "Okay, te puedo abrir una cuenta que puede ser una cuenta de ahorros, una cuenta de plazo fijo, una cuenta sueldo, etcétera". Pero jamás te van a decir solamente: "Te voy a abrir una cuenta". Entonces cuenta también existe como un concepto ya abstracto.

[00:48] ¿Para eso qué vamos a hacer? Vamos a volverla abstracta. Listo. Entonces vemos que sigue compilando el código. Todo bien, guardamos aquí y no tiene ningún tipo de error aquí en el proyecto, vemos que todos los archivos siguen compilando correctamente. Ahora vamos a ver los atributos de la clase abstracta.

[01:16] Vemos que puede tener variables, puede tener campos, si una clase abstracta puede tener campos, puede referenciar objetos no abstractos. Puede, sí. Puede referenciar objetos no abstractos también. Elementos estáticos puede tener, no hay ningún problema, incluso puede tener constructores. Puede tener constructor por defecto. ¿Una clase abstracta será que puede tener un constructor por defecto?

[01:45] Vamos aquí, y también, una clase abstracta puede tener constructor por defecto y constructores personalizados también. En métodos concretos, también, métodos reales, puede tener métodos reales también, puede tener métodos reales y métodos concretos, como el método saca, deposita, etcétera, vemos que él no nos da ningún tipo de problema con esto. Getters y setters también, una clase abstracta también puede tener getters y setters.

[02:17] Si se dan cuenta, el comportamiento sigue siendo el mismo, solamente que ahora ya la clase no puede ser instanciada por sí sola sino que tiene que ser instanciada con una implementación real, una clase que extienda de esta clase abstracta llamada cuenta. Entonces, ya que esa es la única restricción que tenemos con las clases abstractas, para practicar un poco vamos a volver nuestro método deposita abstracto.

[02:50] ¿Y para volverlo abstracto qué necesitamos? Bueno, primero que todo, declararlo abstracto. Pongamos un abstract, deposita, me da error. ¿Por qué me da error? Porque está con la implementación del método. Recordemos que él no debería tener cuerpo, entonces solamente borramos el cuerpo, punto y coma, y ya está nuestro método deposita hecho abstracto.

[03:17] Guardamos aquí y nos saltó dos errores en la cuenta ahorros y la cuenta corriente. Vamos a ver qué errores son. Vamos a cuenta de ahorros. Vinimos aquí, nos está subrayando el error, y nos dice que en efecto el método depositar no está implementado. ¿Qué hacemos? Lo implementamos y aquí está, deposita.

[03:40] Ahora, el método deposita yo no recuerdo cómo era, entonces voy a dar un "Ctrl + Z" para copiar esta línea. Perfecto. Y ahora nuevamente voy a borrar esto y voy a dar punto y coma. Compila aquí, guardo, y aquí en deposita, ¿qué hago? Copio. Y en este caso este objeto de aquí cuenta ahorros no tiene el atributo saldo entonces yo no puedo hacerle un this.saldo.

[04:10] Vamos a repasar también aquí un poco la referencia de this. Este objeto aquí extiende de cuenta pero no tiene saldo. Perfecto. Pero ahora, ¿cuál es el problema? Hay dos problemas. Uno, que yo no tengo ninguna variable llamada saldo aquí, y número dos, que saldo como variable en la clase padre está como privado. ¿Por qué? Vamos a ver aquí. Está private double saldo.

[04:42] Yo podría repetir aquí, declarar digamos un private double saldo, y el código compila perfectamente. Pero estaría repitiendo código que ya existe aquí en la clase padre, entonces eso ya sería un error de diseño. ¿Cómo podría yo solucionar este problema? Primero que todo, este private no me va dejar acceder a saldo por nada del mundo, si es que yo lo vuelvo público o protected.

[05:18] Entonces, si lo vuelvo público él va a ser accesible desde cualquier clase dentro de mi sistema lo cual es muy, muy peligroso. Lo que voy a hacer es volverlo protected. ¿Por qué? Porque con protected, saldo es accesible desde sus clases hijas. Entonces voy a borrar este saldo de aquí que no me va a servir y aquí abajo vemos que él ya accede a this.saldo. ¿Por qué?

[05:44] Porque saldo existe en cuenta y es accesible desde su clase hija cuenta ahorros. Vamos aquí, control, y él nos lleva al atributo de la clase cuenta. Entonces, ya vemos pues que esta clase abstracta, aparte de los atributos que puede tener, nos permite también que las clases hijas accedan a los atributos de las clases padre, como sucedía de igual forma en las clases no abstractas.

[06:14] Ahora, el método abstracto aquí, deposita, solamente puede existir si la clase es abstracta. Por ejemplo, si yo tengo este método aquí, lo voy a copiar por acá, deposita, y mi clase no es abstracta, es una clase normal, entonces me va a dar un error de compilación aquí en cuenta. ¿Por qué? Porque me va a decir que para que yo pueda definir métodos abstractos, mi clase tiene que ser abstracta.

[06:45] Aquí también me está dando el error porque falta el cuerpo de este método y no puede ser abstracto porque mi clase no es abstracta. Entonces

recapitulando un poco aquí en resumen. Clases abstractas, ¿pueden tener campos? Sí. Constructores pueden. Lo único que no pueden tener es una instancia, no puedes instanciar clases abstractas.

[07:11] ¿Métodos abstractos pueden vivir en clases abstractas? Sí. ¿En clases no abstractas? No. Métodos abstractos solamente son posibles en clases abstractas. ¿Las clases normales es opcional que implementen el método abstracto de la clase que extienden? No. Es obligatorio. Toda clase que extienda de una clase abstracta debe implementar los métodos abstractos de esa clase.

[07:42] Entonces nuevamente practiquen mucho, vamos a ver un poco más de clases abstractas en la siguiente clase, y bueno, cualquier duda pueden preguntar en el foro, yo estaré respondiendo ahí. Ya nos estamos viendo. Chau, chau.